IE 305

OPERATİONS RESEARCH II

INSTRUCTOR:CANER TAŞKIN

SİNAN DEMİRHAN

2016402330

**GENERATİNG ITEMS**

int main()

{

int val1[10] ;

int wt1[10] ;

int val2[25] ;

int wt2[25] ;

int val3[100] ;

int wt3[100] ;

int i;

srand(2330);

for(i=0;i<10;i++){

val1[i]=rand()%21 + 1;

wt1[i]=rand()%31+9;

}

for(i=0;i<25;i++){

val2[i]=rand()%21 + 1;

wt2[i]=rand()%31+9;

}

for(i=0;i<100;i++){

val3[i]=rand()%21 + 1;

wt3[i]=rand()%31+9;}

}

**HEURİSTİC APPROACH WHİCH SELECTS THE MOST İMPORTANT VALUES**

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <vector>

#include <string>

#include <iostream>

#include<time.h>

int max(int a, int b) {

return (a > b)? a : b;

}

int knapsack(int W, int wt[], int val[], int n)

{

int i, v1,w1,j;

int a;

int b=0;

int k;

v1=0;

w1=0;

for (i = 0; i <n; i++)

{

for(j=0;j<n;j++){

a=val[j];

if(a>b){

b=a;

k=j;

}

}

if(w1+wt[k]<=W){

w1=w1+wt[k];

v1=v1+b;

}

val[k]=0;

b=0;

}

return v1;

}

int main()

{

int val1[10] ;

int wt1[10] ;

int W1 = 100;

int val2[25] ;

int wt2[25] ;

int W2 = 300;

int val3[100] ;

int wt3[100] ;

int W3 = 750;

int i;

srand(2330);

for(i=0;i<10;i++){

val1[i]=rand()%21 + 1;

wt1[i]=rand()%31+9;

}

for(i=0;i<25;i++){

val2[i]=rand()%21 + 1;

wt2[i]=rand()%31+9;

}

for(i=0;i<100;i++){

val3[i]=rand()%21 + 1;

wt3[i]=rand()%31+9;

}

int n1 = 10;

int n2=25;

int n3=100;

printf("Value1 = %d\n", knapsack(W1, wt1, val1, n1));

printf("Value2 = %d\n", knapsack(W2, wt2, val2, n2));

printf("Value3 = %d", knapsack(W3, wt3, val3, n3));

return 0;

}

**TOP DOWN DYNAMİC PROGRAMMİNG APPROACH**

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <vector>

#include <string>

#include <iostream>

#include<time.h>

//Bottom up

#define max(a,b) (a > b ? a : b)

int topdown(int index, int size, int weights[],int values[]){

int matrix[100][100] = {0};

int take,dontTake;

take = dontTake = 0;

if (matrix[index][size]!=0)

return matrix[index][size];

if (index==0){

if (weights[0]<=size){

matrix[index][size] = values[0];

return values[0];

}

else{

matrix[index][size] = 0;

return 0;

}

}

if (weights[index]<=size)

take = values[index] + topdown(index-1, size-weights[index], weights, values);

dontTake = topdown(index-1, size, weights, values);

matrix[index][size] = max (take, dontTake);

return matrix[index][size];

}

**BOTTOM UP DYNAMİC PROGRAMMİNG APPROACH**

int knapsack(int W, int wt[], int val[], int n)

{

int i, j;

int K[n+1][W+1];

for (i = 0; i <= n; i++)

{

for (j = 0; j <= W; j++)

{

if (i==0 || j==0)

K[i][j] = 0;

else if (wt[i-1] <= j)

K[i][j] = max(val[i-1] + K[i-1][j-wt[i-1]], K[i-1][j]);

else

K[i][j] = K[i-1][j];

}

}

return K[n][W];

}

int main()

{

int val1[10] ;

int wt1[10] ;

int W1 = 100;

int val2[25] ;

int wt2[25] ;

int W2 = 300;

int val3[100] ;

int wt3[100] ;

int W3 = 800;

int i;

srand(2330);

for(i=0;i<10;i++){

val1[i]=rand()%21 + 1;

wt1[i]=rand()%31+9;

}

for(i=0;i<25;i++){

val2[i]=rand()%21 + 1;

wt2[i]=rand()%31+9;

}

for(i=0;i<100;i++){

val3[i]=rand()%21 + 1;

wt3[i]=rand()%31+9;

}

for(i=0;i<10;i++){

printf("%d ",val1[i]);

printf("%d\n",wt1[i]);

}

int n1 = 10;

int n2=25;

int n3=100;

printf("bottom up\n");

printf("Value1 = %d\n", knapsack(W1, wt1, val1, n1));

printf("Value2 = %d\n", knapsack(W2, wt2, val2, n2));

printf("Value3 = %d\n", knapsack(W3, wt3, val3, n3));

printf("top down\n");

printf("Max value1 = %d \n",topdown(n1-1,W1,wt1,val1));

printf("Max value2 = %d \n",topdown(n2-1,W2,wt2,val2));

printf("Max value3 = %d ",topdown(n3-1,W3,wt3,val3));

return 0;

}

**SOLUTİONS**

Taking three different weight capacity for three sets of items.

EXAMPLE 1:W1=75, W2=200, W3=750

EXAMPLE 2:W1=100, W2=350, W3=1000

EXAMPLE 3:W1=150, W2=450, W3=1250

**FOR HEURİSTİC APPROACH**

Solutions for example 1:

Value1 = 73

Value2 = 160

Value3 = 593

Solutions for example 2:

Value1 = 84

Value2 = 230

Value3 = 729

Solutions for example 3:

Value1 = 96

Value2 = 249

Value3 = 839

**FOR BOTTOM UP AND TOP DOWN APPROACH**

Solutions for example 1:

Value1 = 73

Value2 = 172

Value3 = 642

Solutions for example 2:

Value1 = 84

Value2 = 230

Value3 = 779

Solutions for example 3:

Value1 = 96

Value2 = 249

Value3 = 891

**COMPUTATİON TİMES**

**Heuristic approach and bottom up dynaming programming approach lasts in split seconds but in top down dynamic programming approach takes much more times(in minutes) in comparison with the other two approach.**

**COMMENTS**

When I solved the knapsack problem in three different approaches I realized that while capacity is increasing, the difference between heuristic solution and bottom up or top down solutions are also increasing. Also,the quality of heuristic approach that I made is not so good in comparison with bottom up or top down approaches.

For values of size 10……100\*(96-96)/96=0 Good Quality

For values of size 25……100\*(249-249)/249=0 Good Quality

For values of size 100.…100\*(839-891)/891=-0.05 Medium quality

As for the computation times, we can easily see that speed of bottom up approach is much faster than top down approach.